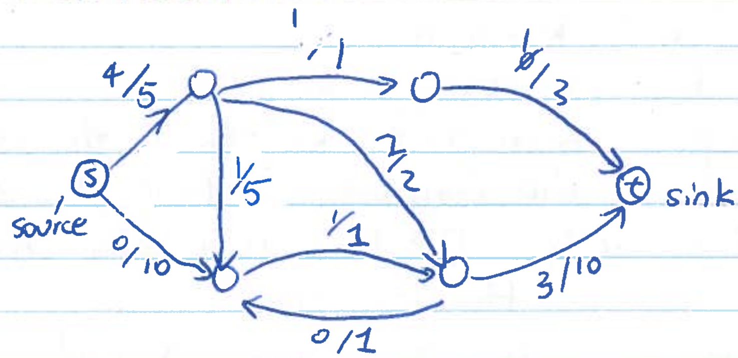
Comp 360

Max-Flow Problem



A flow $f: E \longrightarrow \mathbb{R}$

(i) $0 \leq f(e) \leq c_e$

(ii) $f^{in}(v) = f^{out}(v)$ for every v

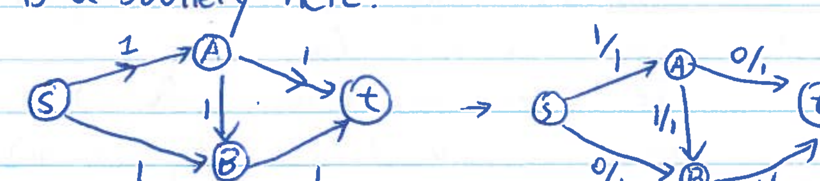
$Val(f) = f^{out}(s)$

max Flow problem, Given a flow network, find the maximum value of a flow.

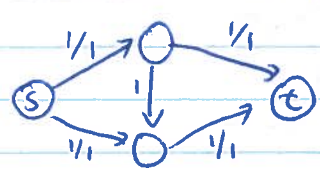
Ford-Fulkerson Algorithm:

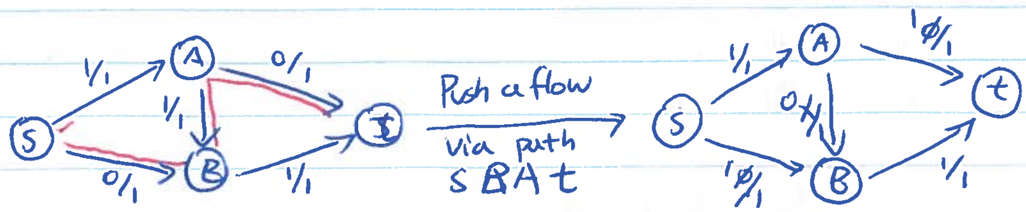
Try to find s-t paths that have not used their Capacity and push more flow through.

There is a subtlety here.



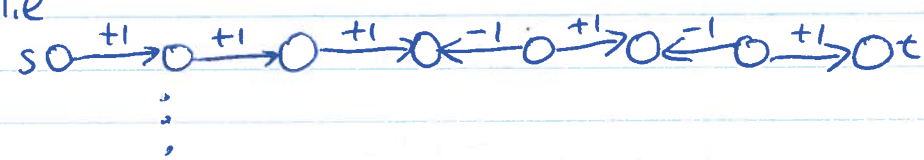
We are stuck. But the flow we get is not optimal.





So here is Ford-Fulkerson Alg:

1. Start from all zero flow
2. Find a parth from s-t such that the edges that are in the forward direction have unused capacity. The backward edges have strictly positive flow on them.
   add one unit to forward edges and subtract one from backward edges.
   i.e

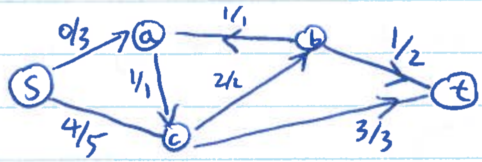


Let's see a definition first

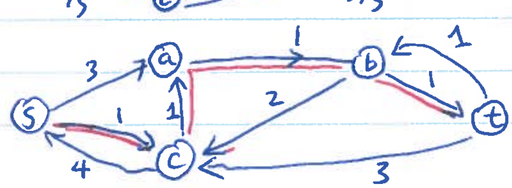
Def: [residual graph] Given a flow network $(G, s, t, \{c_e\})$ and an flow f on G, the residual graph G_f is as follow:

1. Node are the same as G
2. For every edge $uv \in G$ with $f(u,v) < C_{uv}$ add the edge uv with residual capacity $C_{uv} - f(u,v)$ to G_f
3. For every $uv \in G$ with $f(u,v) > 0$ add the opposite edge vu with residual capacity $f(u,v)$

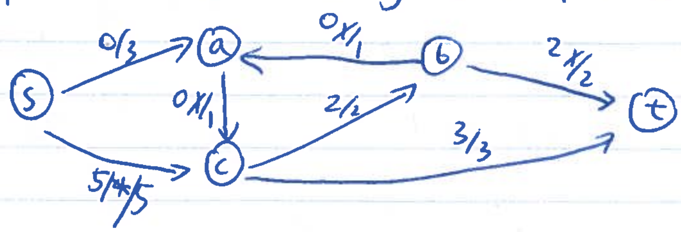
Ex: G:



G_f:



By Following the red parth (s, c, a, b, t)
we push 1 flow through the parth to get

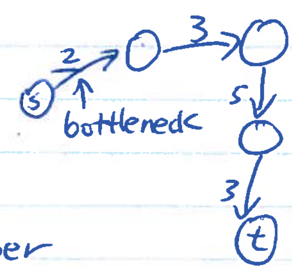


Back to the Ford-Fulkerson algorithm:

① translate to : Set $f(e)=0 \ \forall e \in E$

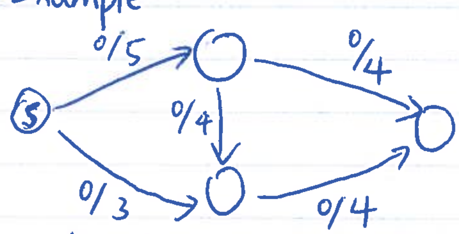
② translate to : - Construct G_f
- while there is an s-t parth P in G_f:
  $f' \leftarrow$ Augment (f, P)
  increase flow vial parth P
  Update $f \leftarrow f'$
  update G_f
  end while.

Augment (f, P)
- Find the bottleneck of P, which is the smallest residual capacity on P.
  For forward edges we add this number to their flow.

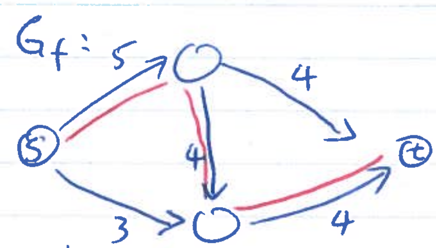


Complete Example:

G:

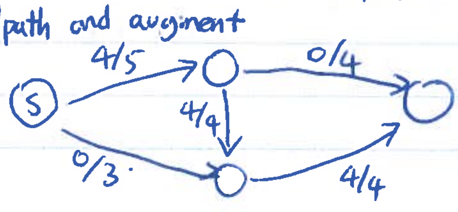


G_f:

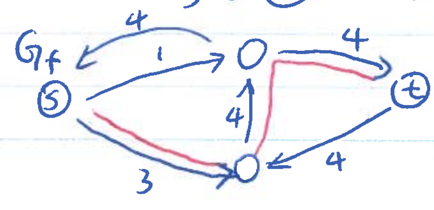


Follow red path and augment we get

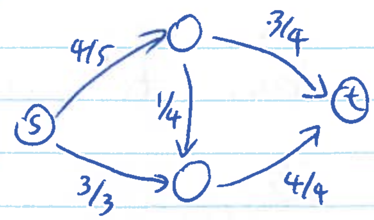
G



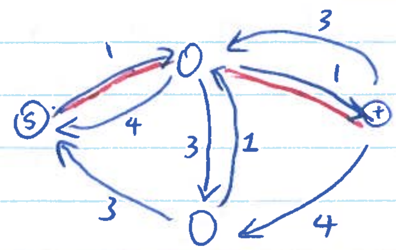
G_f



G

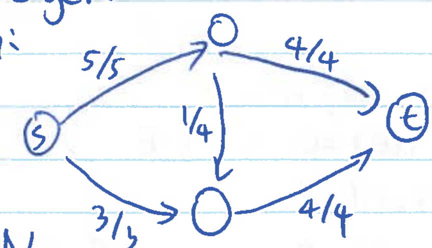


G_f

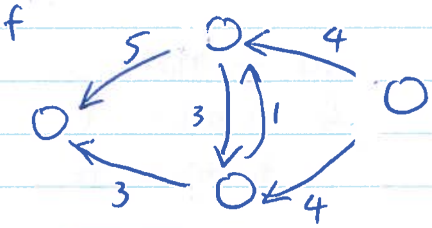


Again, augment via red path we get:

G:



G_f



Now,

No more s-t path in G_f hence we terminate.

Claim: F-F algorithm always return a valid flow.

pf: residual capacities are chosen so that updating with augment(f, P) will never assign a number to an edge that is larger than its capacity or smaller than 0 $\Rightarrow$ Capacity condition is satisfied throughout the algorithm.

Conservation condition:

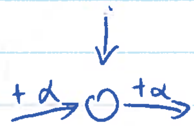
$$f^{in}(v) = f^{out}(v)$$

G_f:

bottleneck $= \alpha$

G

Case 1 :



$f^{in} \leftarrow f^{in} + \alpha$

$f^{out} \leftarrow f^{out} + \alpha$

Still the same

Case 2 :

$$\xrightarrow{+\alpha} (v) \xleftarrow{-\alpha}$$

$$f^{in} = f^{in} + \alpha - \alpha = f^{in}$$

$$f^{out} = f^{out} \qquad \Rightarrow f^{in} = f^{out}$$

Case 3 :

$$\xleftarrow{-\alpha} (v) \xrightarrow{+\alpha}$$

$$f^{in} \leftarrow f^{in}$$

$$f^{out} \leftarrow f^{out} - \alpha + \alpha \qquad \Rightarrow \quad f^{in} = f^{out}$$

Case 4:

$$\xleftarrow{-\alpha} (v) \xleftarrow{-\alpha}$$

$$f^{in} \leftarrow f^{in} - \alpha$$

$$f^{out} \leftarrow f^{out} - \alpha \qquad \Rightarrow \quad f^{in} = f^{out}$$

In all cases $f^{in}(v)$ remains equal to $f^{out}(v)$

Claim : The algorithm terminate.

pf: At every iteration, the flow increases by at least 1 unit. It can never exceed the total sum of all capacity.

Running time: Let K be the largest capacity and
n the number of vertices
m the number of edges.
We have at most $K \cdot m$ iteration.

Each iteration requires DFS. and it takes $O(m+n)$
Also, since we assume every vertex is adjacent to at least one edge $n \geq m/2$, so $O(m+n) = O(m)$
Hence the running time of FF-algorithm is :

$$O(K \cdot m \cdot m) = O(Km^2)$$

Case 2 :

$$\xrightarrow{+\alpha} (v) \xleftarrow{-\alpha}$$

$$f^{in} = f^{in} + \alpha - \alpha = f^{in}$$
$$f^{out} = f^{out} \qquad \Rightarrow f^{in} = f^{out}$$

Case 3 :

$$\xleftarrow{-\alpha} (v) \xrightarrow{+\alpha}$$

$$f^{in} \leftarrow f^{in}$$
$$f^{out} \leftarrow f^{out} - \alpha + \alpha \qquad \Rightarrow f^{in} = f^{out}$$

Case 4 :

$$\xleftarrow{-\alpha} (v) \xleftarrow{-\alpha}$$

$$f^{in} \leftarrow f^{in} - \alpha$$
$$f^{out} \leftarrow f^{out} - \alpha \qquad \Rightarrow f^{in} = f^{out}$$

In all cases $f^{in}(v)$ remains equal to $f^{out}(v)$

Claim : The algorithm terminate.

pf: At every iteration, the flow increases by at least 1 unit. It can never exceed the total sum of all capacity.

Running time : Let K be the largest capacity and
n the number of vertices
m the number of edges.
We have at most $K \cdot m$ iteration.
Each iteration requires DFS. and it takes $O(m+n)$
Also, since we assume every vertex is adjacent to at least one edge $n \geq m/2$, so $O(m+n) = O(m)$
Hence the running time of FF-algorithm is :

$$O(K \cdot m \cdot m) = O(Km^2)$$